



Windows编程 循序渐进

张静盛 编著



附光盘



机械工业出版社
China Machine Press

TP316.7/159D

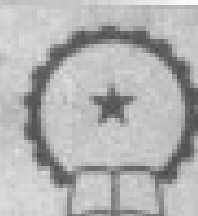
2008

精品系列



Windows编程 循序渐进

张静盛 编著



机械工业出版社
China Machine Press

本书用大量的实例演示使用Visual C++开发Windows应用程序的相关技术。

全书分为3篇19章，分别是软件设计基础篇、软件设计综合应用篇、Windows系统程序设计篇。内容包括软件开发起步、对话框应用程序、基本控件、文档与视图、GDI绘图技术、键盘与鼠标消息、网络通信基础、密码学算法、多媒体技术、数据库技术、综合实例开发、进程与线程、内存管理、进程间通信、线程同步、动态链接库、结构化异常处理、可执行文件格式（PE）、模块注入与函数挂接技术。

本书适用于Windows程序设计的初学者，也可作为大中专院校相关专业教材。另外，本书还适合稍有基础的Visual C++开发者阅读参考。

版权所有，侵权必究。

本书法律顾问 北京市展达律师事务所

图书在版编目（CIP）数据

Windows编程循序渐进 / 张静盛编著. —北京：机械工业出版社，2008.5
(原创精品系列)

ISBN 978-7-111-23862-1

I . W… II . 张… III . 窗口软件，Windows—程序设计 IV . TP316.7

中国版本图书馆CIP数据核字（2008）第049934号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：李东震

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2008年5月第1版第1次印刷

186mm×240mm · 24.75印张

标准书号：ISBN 978-7-111-23862-1

ISBN 978-7-89482-630-5（光盘）

定价：59.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线（010）68326294

前 言

Windows操作系统是一种广泛认可的操作系统，凭借庞大的用户群体和良好的用户体验，一直占据着操作系统领域的主导地位。本书以Windows下系统程序设计技术为主，应用软件开发为辅，向读者展示Windows的系统机制。

本书内容安排

本书设计了大量的实例演示Windows应用程序开发过程中的相关技术，分为3篇。

软件设计基础篇

- 第1章，软件开发起步：编写第一个软件，熟悉MFC应用程序框架。
- 第2章，对话框应用程序：熟悉模态、非模态对话框以及通常对话框的原理与使用方法。
- 第3章，基本控件：介绍按钮、编辑框、列表框等基本控件的使用方法。
- 第4章，文档与视图：介绍文档与视图的基本原理。
- 第5章，GDI绘图技术：介绍GDI绘图技术的基本的GDI对象。
- 第6章，键盘与鼠标消息：介绍键盘、鼠标消息的处理与模拟。

软件设计综合应用篇

- 第7章，网络通信基础：介绍网络模型、协议以及套接字编程和LSP的实现。
- 第8章，密码学算法：介绍常见的密码学算法及其实现。
- 第9章，多媒体技术：介绍几种多媒体控件的使用方式和屏幕截图、录像的实现。
- 第10章，数据库技术：介绍MFC ODBC和DAO基本使用方法。
- 第11章，综合实例开发：实现多个具有趣味性的实例。

Windows系统程序设计篇

- 第12章，进程与线程：介绍进程与线程的原理及其基本应用。
- 第13章，内存管理：介绍虚拟内存与内存映射两种内存管理机制。
- 第14章，进程间通信：介绍共享内存、管道等进程间通信方式的原理与实现方法。
- 第15章，线程同步：介绍多种线程同步技术，包括使用内核对象实现线程同步。
- 第16章，动态链接库：介绍DLL的基本原理，包括TLS机制。
- 第17章，结构化异常处理：介绍结构化异常处理机制，及其在VC++环境下的特性。
- 第18章，可执行文件格式：介绍PE文件格式及其基本应用。
- 第19章，模块注入与函数挂接技术：介绍模块注入及函数挂接技术及其防御。

关于写作本书

编写本书的主要目的是知识总结与经验分享。

对于我来说，知识总结在学习的过程中是必不可少的，就像操作系统需要“磁盘整理”一样。当很长一段时间不对系统进行“磁盘整理”操作，系统性能就会下降。经常总结所学知识，从总体把握体系结构，让自己能够长期保持良好地学习状态。本书实例的一部分来自于学习过程，这里只是按照知识体系把这些实例串联起来。

学习过程重要，知识的交流与分享也同样重要。我曾经在看雪软件安全论坛编程版块建立“Windows系统程序设计”专题，与大家分享学习心得，也有一部分朋友参与到这个专题中。我感觉收获比较大，但意犹未尽，于是就产生编写本书的想法。

现在计算机专业的绝大部分学生就业压力越来越大，归根到底就是不被社会所承认。有些人认为学校所传授的知识与实际应用不接轨，有些人认为许多非计算机专业人员进入这个领域导致竞争激烈。而在我看来，作为计算机专业学生应该尽量寻找主观原因。对于计算机专业学生，在大学4年的自由时代，如何有效利用计算机资源是问题的关键所在。摆在眼前有两个选择：游戏与技术。每个人都能从不同游戏中获取乐趣，但对于技术呢？谁来展现技术的魅力？谁来引导学生对技术的兴趣？这也是编写本书的一个重要原因。希望本书能够引发学生朋友对计算机技术的兴趣，能够使读者在学习技术的过程中获得乐趣，能够从学校走出更多的计算机专业人才。

信息反馈与交流

由于水平有限与时间紧迫，书中难免有所错漏。如果有关于本书的任何问题，欢迎发送电子邮件到zhangjingsheng_nbu@hotmail.com。如果希望交流本书内容，可以发送电子邮件，也可以到看雪软件安全论坛（<http://bbs.pediy.com>）的编程版块进行交流。

致谢

感谢李纲老师，让我明白“学习贵在精而不在多”，使我在程序设计与算法领域打下坚实基础。

感谢徐海峡、郭忠翔，让我明白什么是“能够解决问题的人”，坚定地朝自己的方向前进。

感谢段钢（kanxue），感谢看雪软件安全论坛，论坛是我兴趣的源泉，成长的起点。

感谢213实验室的良师益友，一起学习，一起娱乐，一起成长，让我的大学生活丰富多彩。

感谢我的家人和朋友，你们的支持是对我最大的鼓励。

目 录

前 言

第一篇 软件设计基础篇

第1章 软件开发起步	2
1.1 建立MFC应用程序	2
1.2 分析框架结构	4
1.2.1 框架代码文件的结构	4
1.2.2 应用程序类	5
1.2.3 对话框类	6
1.2.4 添加消息响应	7
第2章 对话框应用程序	9
2.1 模态对话框	9
2.1.1 实例：使用MFC实现模态对话框	9
2.1.2 实例：使用Win32 API实现模态对话框	10
2.2 非模态对话框	12
2.2.1 实例：使用MFC实现非模态对话框	12
2.2.2 实例：使用Win32 API实现非模态对话框	13
2.3 属性对话框	14
2.3.1 实例：多页面切换程序	14
2.3.2 实例：向导对话框	16
2.4 对话框设计技巧	17
2.4.1 控件对齐与排列	17
2.4.2 设置控件逻辑顺序	18
2.5 通用对话框	19
2.5.1 实例：通用“打开”和“另存为”对话框	19
2.5.2 实例：通用“字体”对话框	22
2.5.3 实例：通用“颜色”对话框	23

第3章 基本控件	26
3.1 按钮控件	26
3.1.1 按钮CButton类	26
3.1.2 实例：按钮控件的使用方法	28
3.2 编辑框	30
3.2.1 编辑框CEdit类	30
3.2.2 实例：编辑框的使用方法	32
3.3 列表框	33
3.3.1 列表框CListBox类	33
3.3.2 实例：列表框的使用方法	35
3.4 组合框	36
3.4.1 组合框CComboBox类	37
3.4.2 实例：组合框的使用方法	39
3.5 进度条	41
3.5.1 进度条CProgressCtrl类	41
3.5.2 实例：进度条的使用方法	42
3.6 列表控件	44
3.6.1 列表控件CListCtrl类	44
3.6.2 实例：列表控件的使用方法	45
第4章 文档与视图	47
4.1 文档—视图结构	47
4.1.1 单文档与多文档	47
4.1.2 文档与视图体系	48
4.2 实例：单文档应用程序与文档串行化	52
第5章 GDI绘图技术	57
5.1 图形设备接口GDI	57
5.1.1 设备上下文	57
5.1.2 GDI对象	58
5.1.3 GDI绘图	58
5.2 画笔	58

5.2.1 画笔CPen类	58	7.4.1 实例：TCP服务端和客户端程序 ...	96
5.2.2 实例：使用GDI对象CPen绘图 示例	59	7.4.2 实例：UDP服务器和客户端程序 ...	100
5.3 画刷	60	7.5 实例：使用分层服务提供者LSP截取 网络数据包	103
5.3.1 画刷CBrush类	60	7.5.1 服务提供者接口 (SPI)	103
5.3.2 实例：使用GDI对象CBrush绘图 示例	61	7.5.2 设计实例	103
5.4 位图	63	7.5.3 枚举协议目录	106
5.4.1 位图CBitmap	63	7.5.4 LSP的安装与卸载	108
5.4.2 实例：使用GDI对象CBitmap 示例	64	7.5.5 分层服务提供者 (LSP)	113
第6章 键盘与鼠标消息	67	第8章 密码学算法	118
6.1 键盘消息	67	8.1 数据加密标准 (DES)	118
6.1.1 键盘消息的类型	67	8.1.1 算法描述	118
6.1.2 实例：响应键盘消息示例	68	8.1.2 初始置换与逆初始置换	119
6.1.3 模拟键盘消息	70	8.1.3 生成子密钥	120
6.1.4 实例：模拟键盘消息示例	71	8.1.4 f函数的执行流程	121
6.2 鼠标消息	72	8.1.5 解密过程	122
6.2.1 鼠标消息的类型	72	8.1.6 实例：DES算法加密解密演示 ...	123
6.2.2 实例：处理鼠标消息	73	8.2 国际数据加密算法 (IDEA)	131
6.2.3 实例：模拟鼠标消息	74	8.2.1 算法描述	131
 第二篇 软件设计综合应用篇 		8.2.2 生成子密钥	133
第7章 网络通信基础	80	8.2.3 实例：IDEA算法加密解密演示 ...	134
7.1 网络模型	80	8.3 Blowfish算法	139
7.1.1 OSI参考模型	80	8.3.1 算法描述	139
7.1.2 TCP/IP参考模型	81	8.3.2 生成子密钥和S盒	141
7.2 基础协议	82	8.3.3 实例：Blowfish算法加密 解密演示	141
7.2.1 IP协议	82	8.4 公钥加密算法 (RSA)	146
7.2.2 TCP协议	83	8.4.1 算法描述	146
7.2.3 UDP协议	84	8.4.2 实例：RSA加密解密演示软件 ...	147
7.2.4 ICMP协议	85	第9章 多媒体技术	151
7.3 套接字编程	85	9.1 多媒体控件	151
7.3.1 函数介绍	85	9.1.1 实例：使用Animation控件播放 AVI文件	151
7.3.2 实例：Ping程序	88	9.1.2 实例：使用Windows Media Player 控件播放多媒体文件	152
7.3.3 实例：网络嗅探器	92	9.1.3 实例：使用Real Player控件播放 多媒体文件	153
7.4 服务器与客户端模型	96		

9.2 屏幕截图	154
9.2.1 位图	154
9.2.2 实例：屏幕截图	155
9.3 屏幕录像	157
9.3.1 实现原理	157
9.3.2 实例：屏幕录像	158
第10章 数据库技术	161
10.1 设置ODBC数据源	161
10.1.1 ODBC数据源	161
10.1.2 使用ODBC管理器设置Access 数据源	162
10.2 MFC ODBC数据库编程	163
10.2.1 MFC ODBC概述	163
10.2.2 实例：使用MFC ODBC访问 数据库	164
10.3 MFC DAO数据库编程	169
10.3.1 MFC DAO概述	169
10.3.2 实例：使用MFC DAO访问 数据库	169
第11章 综合实例开发	174
11.1 实例：Huffman编码软件	174
11.1.1 Huffman算法原理	174
11.1.2 具体实现	175
11.2 实例：八数码游戏	178
11.2.1 八数码游戏算法介绍	178
11.2.2 具体实现	179
11.3 实例：游戏寻路算法A*	183
11.3.1 A*算法原理	183
11.3.2 二叉堆在A*中的应用	184
11.3.3 具体实现	186
11.4 实例：“连连看”游戏辅助工具	190
11.4.1 “连连看”算法原理	190
11.4.2 具体实现	191
11.5 实例：“对对碰”游戏辅助工具	196
11.5.1 “对对碰”算法原理	196
11.5.2 具体实现	197
11.6 实例：拼音输入法	199

11.6.1 设计实例	200
11.6.2 拼音字典存储结构——Trie树	200
11.6.3 单字联想	205
11.7 实例：Windows二级文件系统	209
11.7.1 设计实例	209
11.7.2 具体实现	211
11.8 实例：手柄测试器	214
11.8.1 DirectInput手柄输入	214
11.8.2 设计实例	216

第三篇 Windows系统程序设计篇

第12章 进程与线程	222
12.1 进程	222
12.1.1 原理介绍	223
12.1.2 创建进程	223
12.1.3 实例：创建进程	226
12.2 线程	227
12.2.1 原理介绍	227
12.2.2 创建线程	229
12.2.3 实例：创建线程	229
12.3 枚举进程/线程信息	231
12.3.1 实例：使用PSAPI示例	231
12.3.2 实例：使用ToolHelpAPI示例	233
12.3.3 实例：使用Native API示例	235
第13章 内存管理	239
13.1 虚拟内存	239
13.1.1 进程虚拟地址空间	239
13.1.2 实例：查看虚拟内存状态	240
13.1.3 实例：演示虚拟内存的“保留— 提交”特性	243
13.1.4 实例：游戏内存修改器	245
13.2 内存映射文件	249
13.2.1 内存映射文件的原理	249

13.2.2 实例：文件分割器	250	机制	287
第14章 进程间通信	254	15.6 信标内核对象	288
14.1 消息传递机制	254	15.6.1 基本原理	288
14.1.1 消息传递	254	15.6.2 实例：使用信标内核对象示例	289
14.1.2 实例：使用WM_COPYDATA 消息传递数据	254	15.7 互斥内核对象	291
14.2 共享内存	256	15.7.1 基本原理	292
14.2.1 共享内存的原理	256	15.7.2 实例：使用互斥内核对象示例	292
14.2.2 实例：使用共享内存示例	257	第16章 动态链接库	295
14.3 管道和邮槽	259	16.1 DLL基础	295
14.3.1 管道和邮槽通信原理	259	16.1.1 DLL的隐式链接	295
14.3.2 实例：使用匿名管道重定向 程序输出	261	16.1.2 DLL的显式加载	296
14.3.3 实例：命名管道示例	263	16.2 编写动态链接库	297
14.3.4 实例：邮槽通信示例	266	16.2.1 入口函数DllMain	297
14.4 剪贴板	267	16.2.2 实例：编写DLL实现导出变量、 函数、类	298
14.4.1 剪贴板通信机制	267	16.3 线程本地存储器 (TLS)	301
14.4.2 实例：使用剪贴板实现进程间 通信示例	269	16.3.1 静态TLS和动态TLS	301
第15章 线程同步	275	16.3.2 实例：使用静态TLS示例	303
15.1 原子访问	275	16.3.3 实例：使用动态TLS示例	304
15.1.1 多线程访问共享数据问题	275	第17章 结构化异常处理	306
15.1.2 互锁系列函数	276	17.1 SEH的概念、特性	306
15.2 关键代码段	277	17.2 SEH的基本使用方法	307
15.2.1 基本原理	277	17.2.1 结束异常程序	307
15.2.2 实例：多线程环境下的数据 共享	278	17.2.2 异常处理程序	310
15.3 内核对象与等待函数	280	17.2.3 顶层异常处理	313
15.3.1 内核对象	280	17.3 VC++编译器级SEH的具体实现	313
15.3.2 等待函数	281	17.3.1 SEH相关数据结构的介绍	314
15.4 事件内核对象	283	17.3.2 异常处理链结构图	315
15.4.1 基本原理	283	17.3.3 实例：单嵌套异常块演示程序	316
15.4.2 实例：使用事件内核对象示例	284	17.3.4 实例：多嵌套异常块演示程序	318
15.5 等待定时器内核对象	285	17.3.5 VC++编译器级异常帧结构	320
15.5.1 基本原理	285	17.3.6 VC中的顶层异常处理	320
15.5.2 实例：使用等待定时器的APC 机制	287	17.3.7 VC搜索异常处理程序流程	322
		第18章 可执行文件格式	324
		18.1 PE文件格式	324
		18.1.1 PE文件头	324
		18.1.2 可选文件头	325

18.1.3 区块表	327	19.2.3 古老的API HOOK	353
18.1.4 输入表	328	19.2.4 实例：HOOK API示例	354
18.1.5 输出表	329	19.2.5 Detours Hook	356
18.1.6 资源表	330	19.2.6 实例：使用detour库实现挂接 API示例	357
18.1.7 重定位表	332	19.3 钩子	359
18.1.8 绑定输入表	332	19.3.1 钩子的基本原理	359
18.2 综合应用	333	19.3.2 钩子类型	360
18.2.1 实例：PE文件资源查看器	333	19.3.3 实例：全局鼠标钩子示例	366
18.2.2 实例：为应用程序添加Nag窗口	337	19.3.4 实例：全局键盘钩子示例	369
第19章 模块注入与函数挂接技术	341	19.3.5 实例：使用局部CBT钩子示例	370
19.1 模块注入	341	19.3.6 实例：使用低级键盘钩子示例	371
19.1.1 添加导入表项	342	19.4 反注入技术	372
19.1.2 远程线程技术	344	19.4.1 实例：使用调试钩子屏蔽全局 钩子	372
19.1.3 实例：使用远程线程实现模块 注入	345	19.4.2 实例：检测注入模块	374
19.1.4 异步过程调用（APC）	346	19.4.3 实例：使用DLL_THREAD_ATTACH 阻止远程线程	377
19.1.5 实例：使用APC实现模块注入	347	19.4.4 实例：使用挂钩LoadLibraryExW 屏蔽全局钩子	379
19.2 挂接API	349	附录 光盘源码实例	381
19.2.1 重定向API	350		
19.2.2 实例：重定向API MessageBoxA 示例	350		

第一篇

软件设计基础篇

本篇主要内容：

- ☐ 第1章 软件开发起步
- ☐ 第2章 对话框应用程序
- ☐ 第3章 基本控件
- ☐ 第4章 文档与视图
- ☐ 第5章 GDI绘图技术
- ☐ 第6章 键盘与鼠标消息

第1章 软件开发起步

从程序设计转化为软件设计，这是激动人心的时刻。这里面对的不再是黑色的、单调的控制台界面，而是一个崭新的环境。各种功能强大、界面漂亮的软件，将从你的手中诞生。软件开发的第一步是熟悉开发环境。本章将会介绍软件基本的开发框架，并对这个框架进行详细的介绍。

1.1 建立MFC应用程序

本节主要介绍如何使用MFC框架生成默认的对话框软件。作为第一个软件，很简单，也不用添加任何代码，权当练手，熟悉一下开发环境。

启动Visual Studio 2005后，首先需要新建项目，开发环境会针对特定的项目类型生成相应的框架代码。可以使用菜单命令“文件→新建→项目”，也可以直接使用快捷键Ctrl+Shift+N。打开“新建项目”对话框，如图1-1所示。

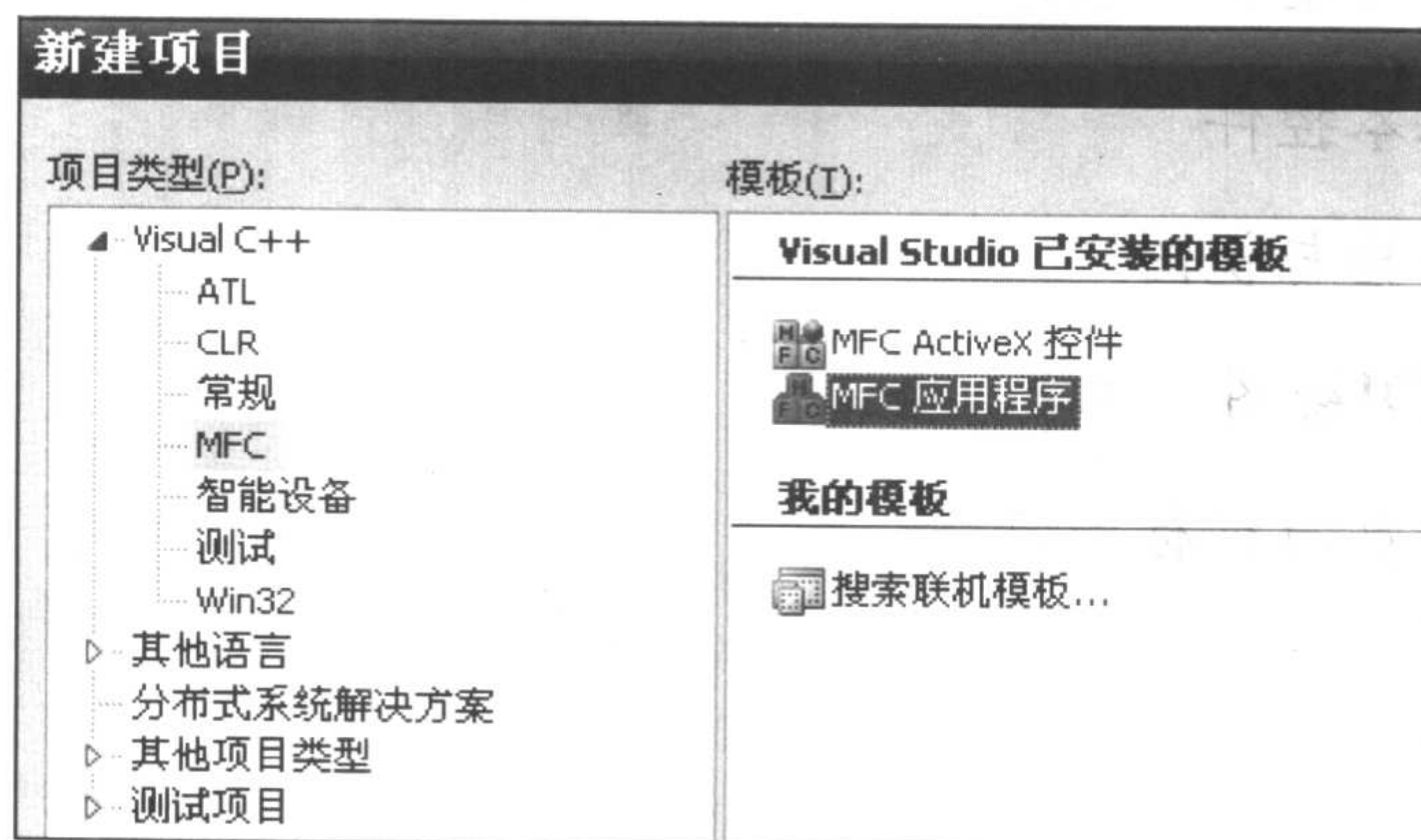


图1-1 Visual Studio 2005的“新建项目”对话框

双击选择“MFC应用程序”类型，输入项目名称、选择项目位置，点击“确定”按钮。出现“MFC应用程序向导”对话框，切换到“应用程序类型”页，如图1-2所示。

选择“基于对话框”应用程序类型，设置“在静态库中使用MFC”，点击“完成”按钮。此时，开发环境就会自动生成基于对话框应用程序的MFC框架代码。

设置为“在静态库中使用MFC”是因为Visual Studio 2005生成的软件会带有MFC80U.dll，这就使得所生成的软件只能在安装有Visual Studio 2005的计算机上使用，而无法在普通计算机上应用。对于初学者建议使用“在静态库中使用MFC”选项。

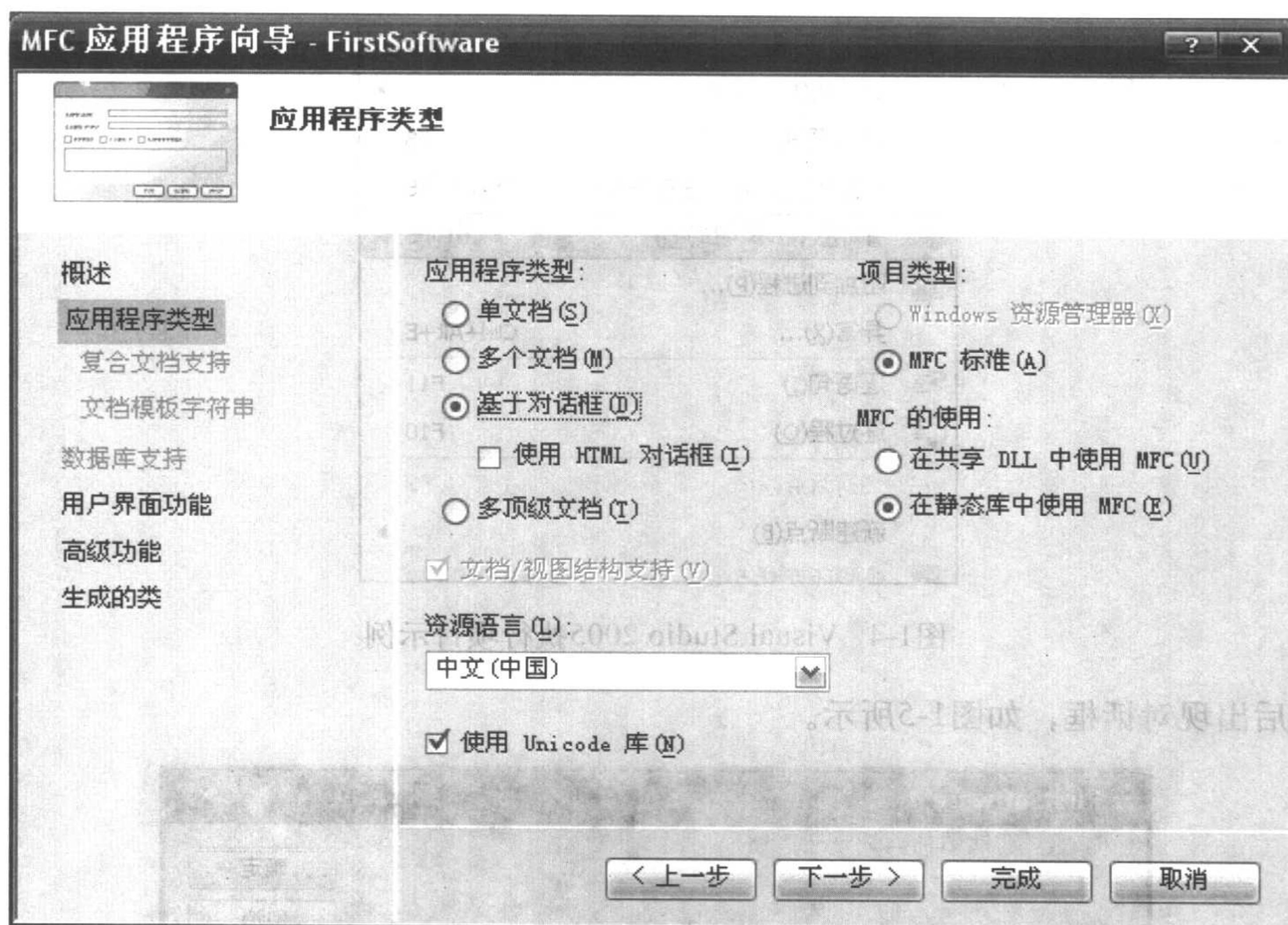


图1-2 Visual Studio 2005 “MFC应用程序向导”对话框

接下来就直接编译框架代码，可以使用菜单命令“生成→生成解决方案”，或者使用快捷键F7，如图1-3所示。编译完成后，执行软件，可以使用菜单命令“调试→开始执行（不调试）”，也可以使用快捷键Ctrl+F5，如图1-4所示。

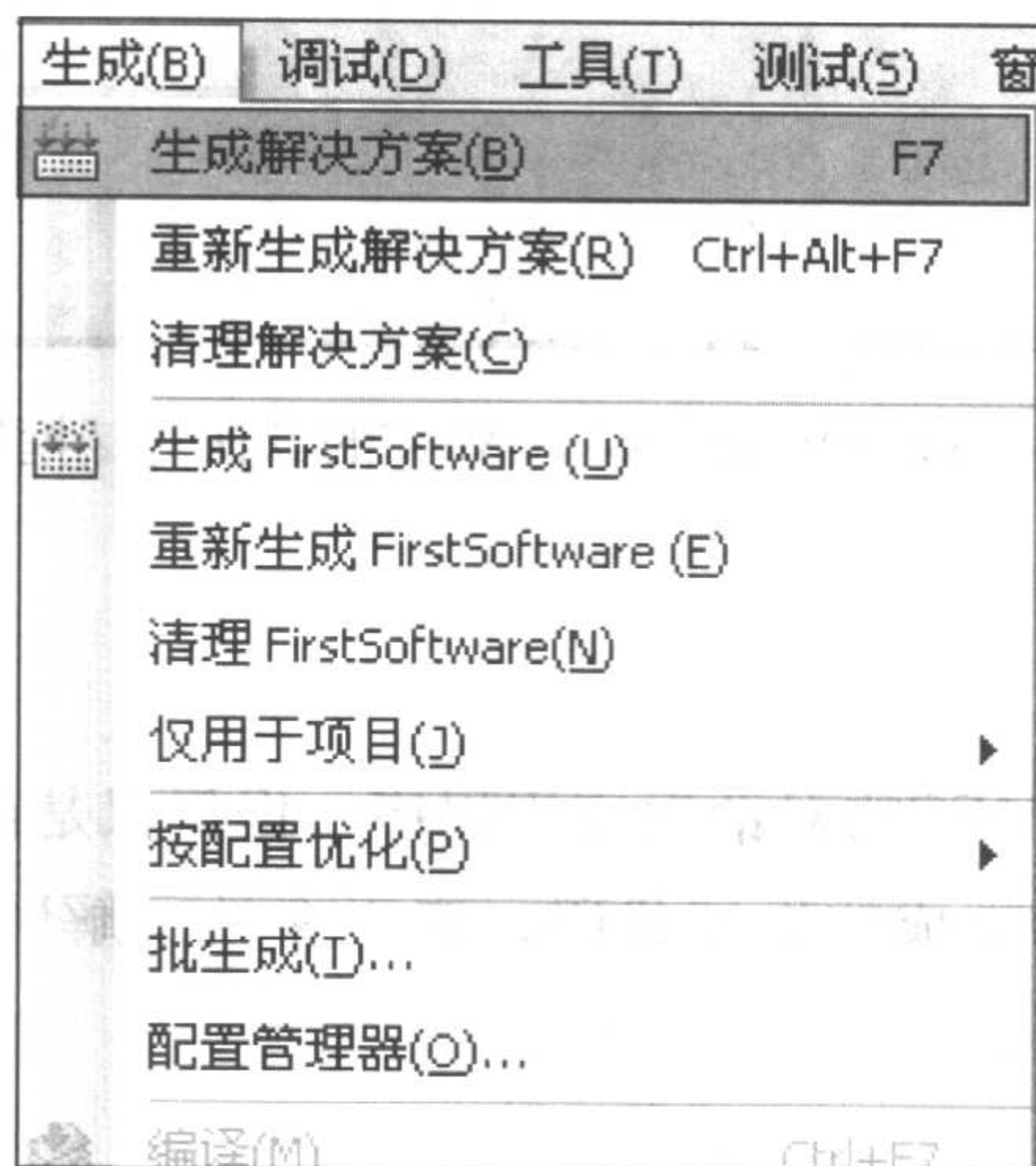


图1-3 Visual Studio 2005编译项目示例

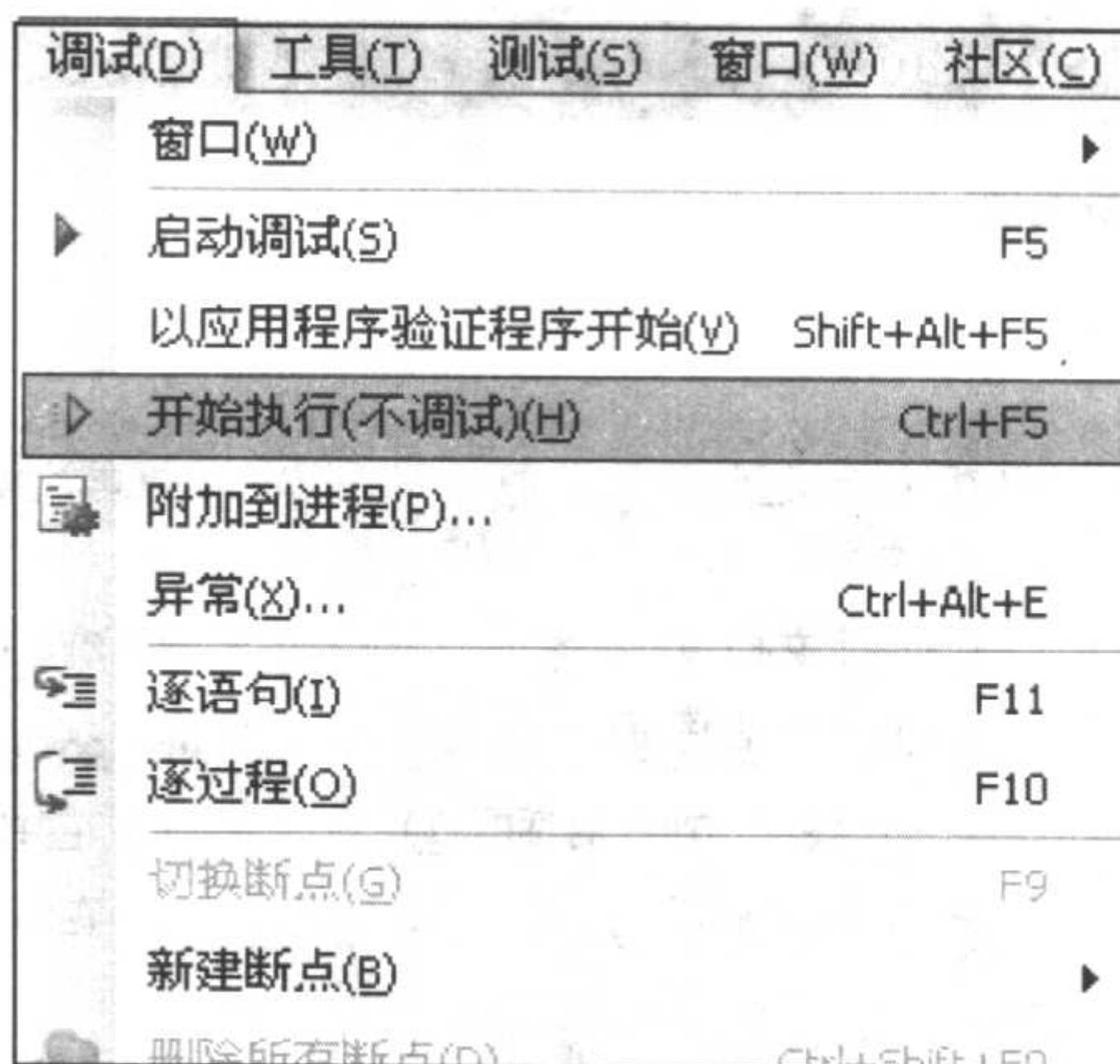


图1-4 Visual Studio 2005执行项目示例

执行后出现对话框，如图1-5所示。

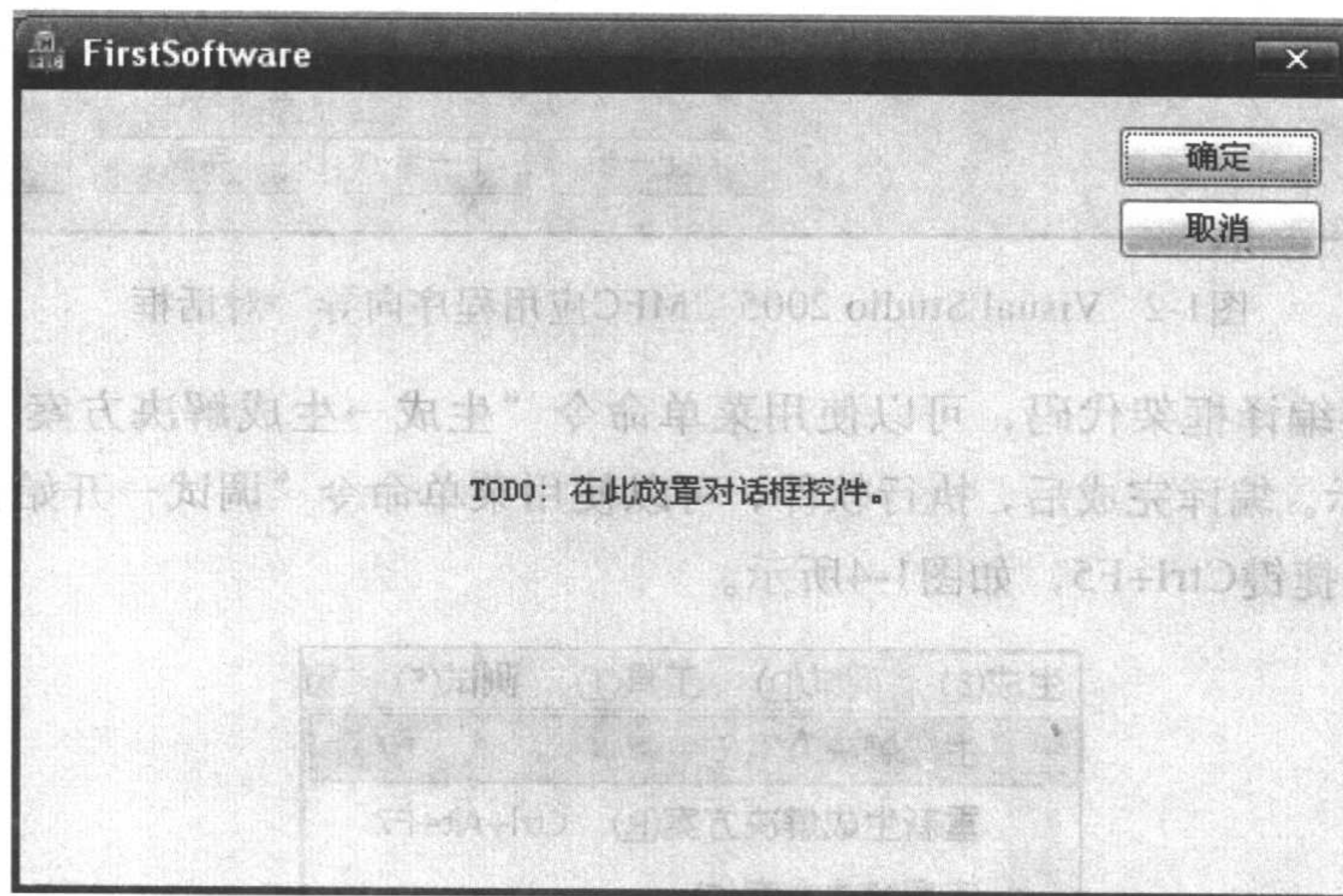


图1-5 Visual Studio 2005基于MFC的默认对话框软件示例

1.2 分析框架结构

在上一节已经生成了基于MFC的对话框应用程序，但仅仅是简单的操作，并没有涉及任何代码。本节将要介绍上一节所生成的框架代码，来更多地了解MFC应用程序，能够更好地在MFC框架快速地开发软件。

1.2.1 框架代码文件的结构

基于MFC对话框程序的框架代码主要由以下几个部分组成：

- 应用程序类：项目名称为FirstSoftware，对应类名为CFirstSoftwareApp。
- 对话框类：项目名称为FirstSoftware，对应类名为CFirstSoftwareDlg。
- 资源文件：定义资源ID，一般不用手动修改。
- 预编译文件：可以用来解决头文件包含冲突的问题，定义一些需要全局性包含的文件。

1.2.2 应用程序类

MFC定义了一个应用程序基类CWinApp，所有基于MFC的应用程序都会继承这个类。FirstSoftware项目也不例外，此时的应用程序类是CFirstSoftwareApp，定义如下：

```
class CFirstSoftwareApp : public CWinApp
{
public:
    CFirstSoftwareApp();

    // 重载虚函数
public:
    virtual BOOL InitInstance();

    // 消息映射
    DECLARE_MESSAGE_MAP()
};
```

CSoftwareApp类定义很简单，重点来看一下InitInstance函数：

```
BOOL CFirstSoftwareApp::InitInstance()
{
    // 初始化操作，省略部分代码……
    CWinApp::InitInstance();
    // 省略部分代码……
    // 定义对话框对象
    CFirstSoftwareDlg dlg;
    // 保存对话框到全局变量
    m_pMainWnd = &dlg;
    // 显示对话框
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // .....
    }
    else if (nResponse == IDCANCEL)
    {
        // .....
    }
    return FALSE;
}
```


姑且可以把应用程序类的这个InitInstance成员函数看作MFC程序的入口点，这里首先会初始化应用程序环境包括控件等，然后再启动对话框。在实际开发过程一般不需要对这个类进行操作，但如果需要在建立对话框之前处理某些数据或者做一些准备工作，那么就可以把代码添加到InitInstance启动对话框之前。

1.2.3 对话框类

MFC对话框类CFirstSoftwareDlg继承CDialog类，负责与用户的交互，处理用户消息，接受用户输入。类定义如下：

```
class CFirstSoftwareDlg : public CDialog
{
public:
    // 构造函数
    CFirstSoftwareDlg(CWnd* pParent = NULL);

    // 对话框ID
    enum {IDD = IDD_FIRSTSOFTWARE_DIALOG};

protected:
    // 动态数据交换，负责控件与变量之间的关联
    virtual void DoDataExchange(CDataExchange* pDX);

protected:
    // 应用程序图标句柄
    HICON m_hIcon;

    // 重载初始化对话框
    virtual BOOL OnInitDialog();

    // 定义消息WM_SYSCOMMAND处理函数
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);

    // 定义消息WM_PAINT处理函数
    afx_msg void OnPaint();

    // 定义消息WM_QUERYDRAGICON处理函数
    afx_msg HCURSOR OnQueryDragIcon();

    // 消息映射
    DECLARE_MESSAGE_MAP()
};
```

从这个类的定义中可以看出以下两点：

- 控件与数据的关联，可以简单地交给框架来实现。
- 在MFC框架上开发主要是针对消息处理机制。

1.2.4 添加消息响应

本小节演示如何在FirstSoftware对话框上添加一个消息响应。首先把工作区切换到资源视图，也可以通过菜单命令“视图→资源视图”来切换，如图1-6所示。

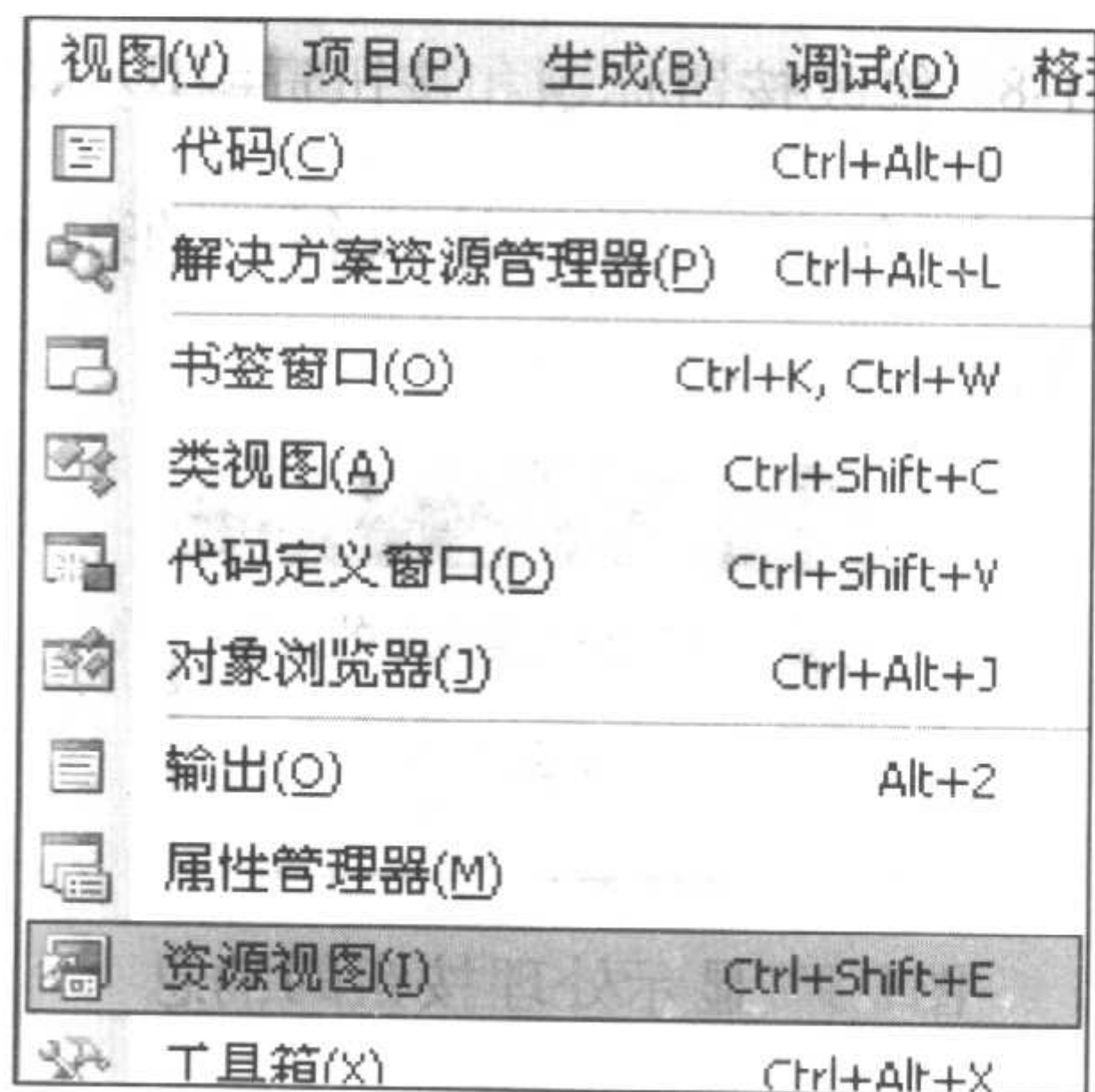


图1-6 Visual Studio 2005打开资源视图示例

切换到资源视图后找到对话框IDD_FIRSTSOFTWARE_DIALOG，如图1-7所示。



图1-7 MFC对话框程序资源视图示例

双击该选项后就会出现可编辑的对话框窗口，从工具栏中选择“Button”按钮后，就可以在对话框窗口拖放该按钮，用鼠标右击按钮，在出现的菜单中选择“属性”切换到属性页，修改按钮名称为“First Button”，修改按钮ID为IDC_FIRST_BUTTON，如图1-8所示。

双击该按钮就会来到按钮的消息处理函数，如下所示：

```
// 框架添加的按钮IDC_FIRST_BUTTON的单击消息处理函数
void CFirstSoftwareDlg::OnBnClickedFirstButton()
{
    // TODO: 在此添加控件通知处理程序代码
}
```



```
this->MessageBox(L"这是我添加按钮消息后的结果!");  
}
```

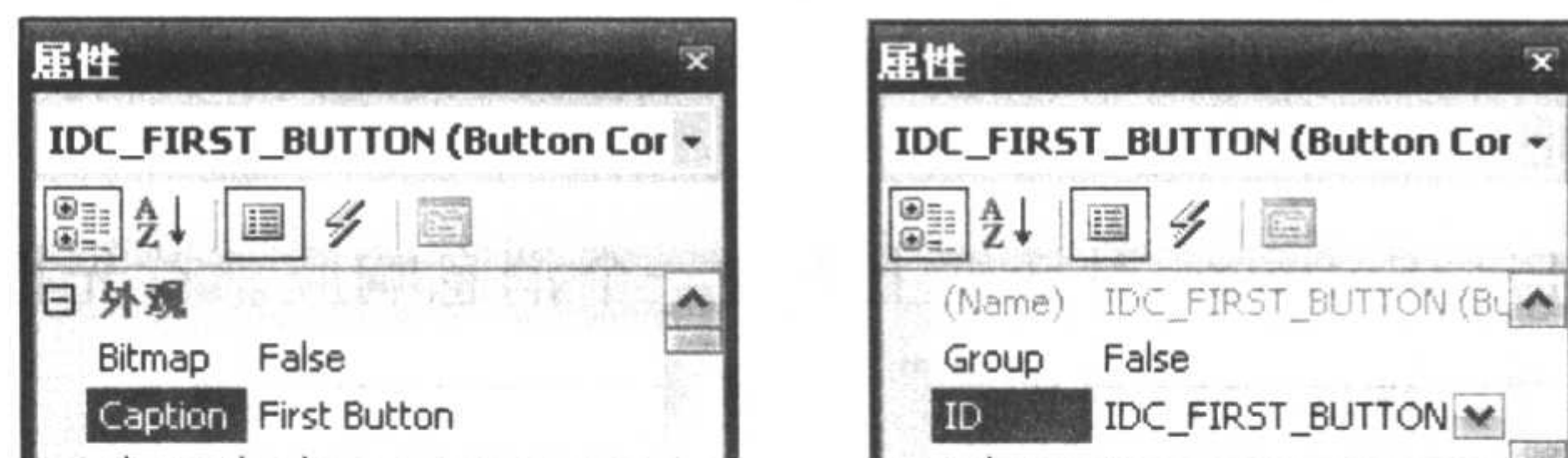


图1-8 修改按钮标题和按钮资源ID示例

在其中添加一个MessageBox语句后，编译并执行。在软件中点击我们所添加的“First Button”按钮后，出现消息框，如图1-9所示。

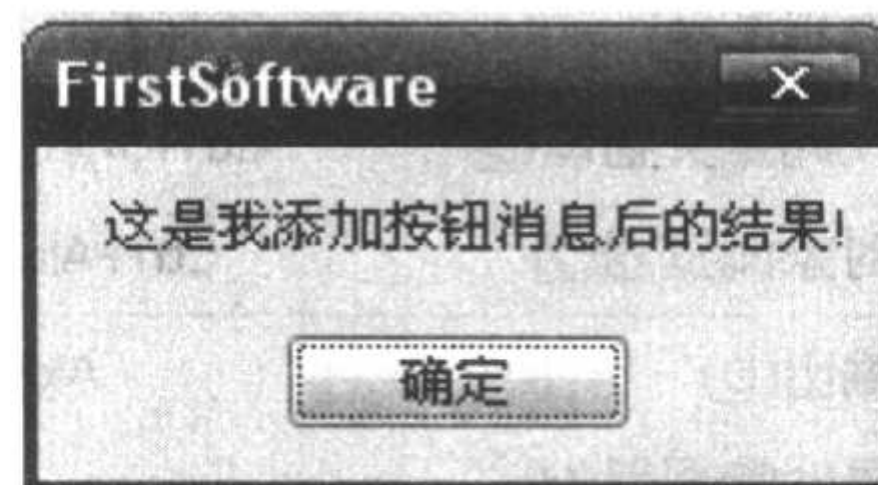


图1-9 显示处理按钮的消息

第2章 对话框应用程序

设计对话框应用程序一般比较简单。对话框窗体包括在对话框上的普通控件，都可以直接在资源编辑器中进行编辑。本章首先介绍几种基本对话框的原理和实现方法，然后介绍一些实用的对话框应用程序设计技巧。

2.1 模态对话框

Windows是建立在消息驱动机制上的，模态对话框正是利用对Windows消息的特殊处理方式，才实现了其特有的效果。

大部分人应该都遇到过这样的情况，例如在使用Word编辑文章时，打开“字体”对话框后，再用鼠标去选择应用程序的其他部分只会听到“嘟嘟”声，这是因为“字体”对话框是一个模态对话框。当打开一个模态对话框时，用户只能与该模态对话框进行交互，而应用程序的其他界面都得不到用户输入的信息。

为什么会有这样的效果呢？其实也不难分析，由于Windows是消息驱动机制的，也就是说只要有有了消息，应用程序肯定会处理对应的事务。当模态对话框产生后，应用程序没有处理消息，唯一的解释就是模态对话框产生后会屏蔽其他窗体消息。事实正是如此，创建模态对话框后，应用程序只会响应该对话框的消息，直到应用收到结束模态对话框窗体的消息后，才会把控制权交还给应用程序。

2.1.1 实例：使用MFC实现模态对话框

本小节就介绍如何使用MFC的CDialog实现模态对话框。实现步骤如下：

- 1) 使用应用程序向导创建基于对话框的MFC程序。
- 2) 使用资源编辑器，在主界面添加一个按钮“IDC_CREATE_DIALOG”用来创建模态对话框。
- 3) 在“资源视图”中添加一个对话框，ID为“IDD_MODAL_DIALOG”。
- 4) 为这个新对话框添加基于CDialog类的CMyModalDialog类。
- 5) 为“IDC_CREATE_DIALOG”添加响应，代码如下：

```
CMyModalDialog dlg ;  
dlg.DoModal();
```

执行实例后，效果如图2-1所示。

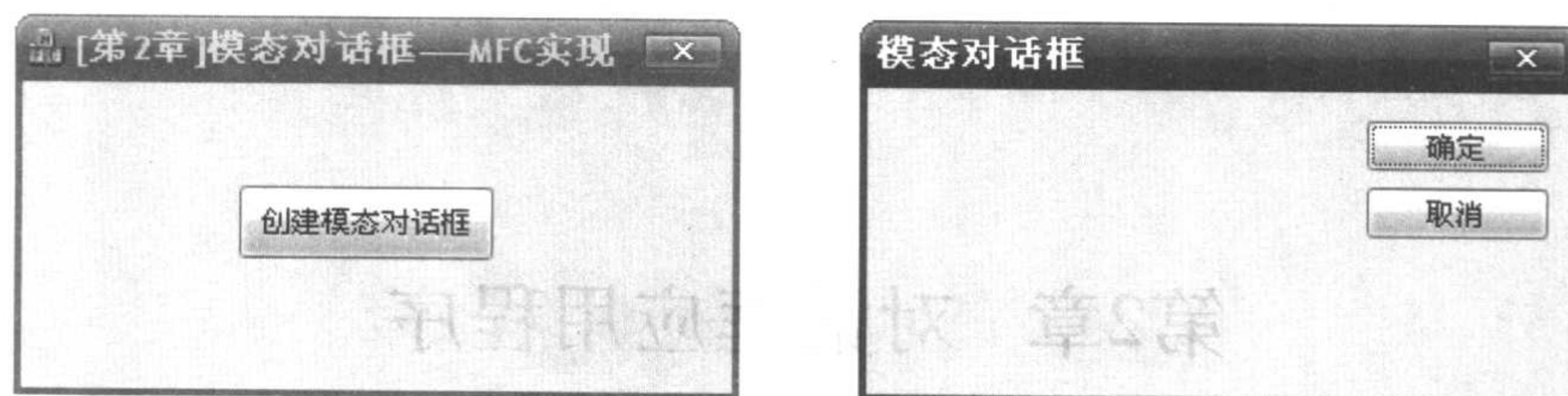


图2-1 模态对话框之MFC实现效果

点击图2-1左对话框的按钮后，就会创建右边的对话框。此时，左对话框是无法接受消息进行处理的，应用程序的焦点始终都在模态对话框上，只有模态对话框结束后左对话框才能恢复焦点。

2.1.2 实例：使用Win32 API实现模态对话框

通过上一小节可以看到，使用MFC的CDialog类实现模态对话框非常方便，然而对于初学者，理解对话框实现机理没有任何帮助。这里因为MFC已经封装很深，很多细节都在其内部实现。本小节就来学习如何用WIN32 API实现模态对话框，可以接触更多的实现细节。

首先接受几个与模态对话框相关的API：

1) DialogBox函数。

```
INT_PTR DialogBox(
    HINSTANCE hInstance,           // 资源所在模块的实例句柄
    LPCTSTR lpTemplate,           // 资源模板名称
    HWND hWndParent,              // 父窗口句柄
    DLGPROC lpDialogFunc          // 对话框的消息处理过程
);
```

2) DialogBoxParam函数，与DialogBox相比唯一的区别多了最后一个参数dwInitParam。

```
INT_PTR DialogBoxParam(
    HINSTANCE hInstance,           // 资源所在模块的实例句柄
    LPCTSTR lpTemplateName,       // 资源模板名称
    HWND hWndParent,              // 父窗口句柄
    DLGPROC lpDialogFunc,         // 对话框的消息处理过程
    LPARAM dwInitParam            // 指定传递给WM_INITDIALOG消息的lParam参数
);
```

这里有3个参数需要说明。

- hInstance：一般情况下，资源和主程序都是在同一模块。然而在实际软件开发中，经常会把资源集中放到一个DLL中，此时就需要先取得资源所在模块的实例句柄。
- lpTemplate：资源模版名称，可以使用MAKEINTRESOURCE宏来转换资源ID。
- hWndParent：父窗口句柄。如果设置为非NULL，就表示创建模态对话框，否则就表示创建非模态对话框。

DialogProc对话框消息处理过程定义如下：

```
INT_PTR CALLBACK DialogProc(  
    HWND hwndDlg,           // 对话框句柄  
    UINT uMsg,              // 消息类型  
    WPARAM wParam,          // 消息的wParam参数  
    LPARAM lParam           // 消息的lParam参数  
);
```

下面就来介绍用WIN32 API创建模态对话框的具体实现方法：

- 1) 使用应用程序向导创建基于对话框的MFC程序。
- 2) 使用资源编辑器，在主界面添加一个按钮“IDC_CREATE_DIALOG”用来创建模态对话框。
- 3) 在“资源视图”中添加一个对话框，ID为“IDD_MODAL_DIALOG”。
- 4) 定义对话框的消息处理过程如下：

```
INT_PTR CALLBACK DialogProc( HWND hwndDlg, UINT uMsg, WPARAM wParam, LPARAM lParam )  
{  
    switch(uMsg)  
    {  
        // 处理对话框初始化消息  
        case WM_INITDIALOG:  
            // 此时的lParam就是DialogBoxParam的最后一个参数dwInitParam所设置的值  
            return TRUE ;  
        // 处理WM_COMMAND消息  
        case WM_COMMAND:  
            {  
                switch(LOWORD(wParam))  
                {  
                    // 响应对话框的结束消息  
                    case IDCANCEL:  
                        ::EndDialog(hwndDlg, 0);  
                        return TRUE;  
                }  
            }  
            break;  
    }  
    // 返回FALSE，表示把消息继续传递给系统默认的消息处理过程  
    // 返回TRUE，就表示不需要继续传递消息  
    return FALSE;  
}
```

- 5) 为按钮“IDC_CREATE_DIALOG”添加消息响应。

```
// 取得当前模块的实例句柄  
HINSTANCE hInstance = (HINSTANCE)GetModuleHandle(NULL);
```